

Tensor machines for learning target-specific polynomial features

Jiyan Yang¹ and Alex Gittens²

¹ Stanford University
Stanford, CA, USA
jiyan@stanford.edu,

² International Computer Science Institute
Berkeley, CA, USA
gittens@icsi.berkeley.edu

Abstract. Recent years have demonstrated that using random feature maps can significantly decrease the training and testing times of kernel-based algorithms without significantly lowering their accuracy. Regrettably, because random features are target-agnostic, typically thousands of such features are necessary to achieve acceptable accuracies. In this work, we consider the problem of learning a small number of explicit polynomial features. Our approach, named *Tensor Machines*, finds a parsimonious set of features by optimizing over the hypothesis class introduced by Kar and Karnick for random feature maps in a *target-specific* manner. Exploiting a natural connection between polynomials and tensors, we provide bounds on the generalization error of Tensor Machines. Empirically, Tensor Machines behave favorably on several real-world datasets compared to other state-of-the-art techniques for learning polynomial features, and deliver significantly more parsimonious models.

1 Introduction

Kernel machines are one of the most popular and widely adopted paradigms in machine learning and data analysis. This success is due to the fact that an appropriately chosen non-linear kernel often succeeds in capturing non-linear structures inherent to the problem without forming the high-dimensional features necessary to explicitly delineate those structures. Unfortunately, the cost of kernel methods scales like $\mathcal{O}(n^3)$, where n is the number of datapoints. Because of this high computational cost, it is often the case that kernel methods cannot exploit all the information in large datasets; indeed, even the $\mathcal{O}(n^2)$ cost of forming and storing the kernel matrix can be prohibitive on large datasets.

As a consequence, methods for approximately fitting non-linear kernel methods have drawn much attention in recent years. In particular, explicit random feature maps have been proposed to make large-scale kernel machines practical [14,7,13,6,22,23]. The idea behind this method is to randomly choose an r -dimensional feature map ϕ that satisfies $k(\mathbf{x}, \mathbf{y}) = \mathbb{E} [\phi(\mathbf{x})^T \phi(\mathbf{y})]$, where k is the kernel of interest [14]. Let Φ be the matrix whose rows comprise the

application of ϕ to n datapoints, then the kernel matrix \mathbf{K} has the low-rank approximation $\Phi\Phi^T$. This approximation considerably reduces the costs of both fitting models, because one can work with the n -by- r matrix Φ instead of \mathbf{K} , and of forming the input, because one need form only Φ instead of \mathbf{K} .

The success of this approximation hinges on choosing ϕ so that $\phi(\mathbf{x})^T\phi(\mathbf{y})$ is close to $k(\mathbf{x},\mathbf{y})$ with high probability. One easy way to accomplish this is to take $\phi(\mathbf{x}) = \frac{1}{\sqrt{r}}(\phi_1(\mathbf{x}), \dots, \phi_r(\mathbf{x}))$, where the random features ϕ_i all satisfy $\mathbb{E}[\phi_i(\mathbf{x})\phi_i(\mathbf{y})] = k(\mathbf{x},\mathbf{y})$. The concentration of measure phenomenon then ensures that $|k(\mathbf{x},\mathbf{y}) - \phi(\mathbf{x})^T\phi(\mathbf{y})|$ is small with high probability. In practice, one must use a large number of random features to achieve performance comparable with the exact kernel: r must be on the order of tens or even hundreds of thousands [6]. The reason for this is simple: since knowledge of the target function is not used in generating the feature map ϕ , to ensure good performance enough random features must be selected to get good approximations to *arbitrary* functions in the reproducing kernel Hilbert space associated with k . Thus under the random feature map paradigm, one trades off a massive reduction in the cost of optimization for the necessity of generating a large number of random features, each of which is rather uninformative about any given target.

Implicit in the formulation of random feature maps is the assumption that the chosen class of features is expressive enough to cover the function space of interest, yet simple enough that features can be sampled randomly in an efficient manner. Given this assumption, it seems natural to question whether it is possible to directly select a much smaller number of features relevant to the given target in a computational efficient manner.

Target-specific optimization vs target-agnostic randomization

Given a kernel k and a parametrized family of features $\{\phi_\omega\}_{\omega \in \Omega}$ satisfying $k(\mathbf{x},\mathbf{y}) = \mathbb{E}[\phi_\omega(\mathbf{x})^T\phi_\omega(\mathbf{y})]$ where the expectation is with respect to some distribution on Ω , can optimization over this hypothesis class to select r features give a parsimonious representation of a known target, with the same or less computational effort and less error than target-agnostic random sampling of ω ?

This question has recently been answered in the affirmative for the case of the radial basis kernel [24,25]. In this work, we provide a positive answer to the above question in the case of polynomial kernels and the Kar–Karnick random features introduced in [7] (see Figure 1). Polynomial kernels are of interest because, after the radial basis kernel, they are arguably the most widely used kernels in machine learning applications. The Weierstrass approximation theorem guarantees that any smooth function can be arbitrarily accurately approximated by a polynomial [17][Chapter 7], so in principle a polynomial kernel of sufficiently high degree can be used to accurately approximate any smooth target function.

In this work, we use a natural connection between tensors and polynomials to introduce a restricted hypothesis class of polynomials corresponding to low-rank tensors. In the Tensor Machine paradigm, learning consists of learning

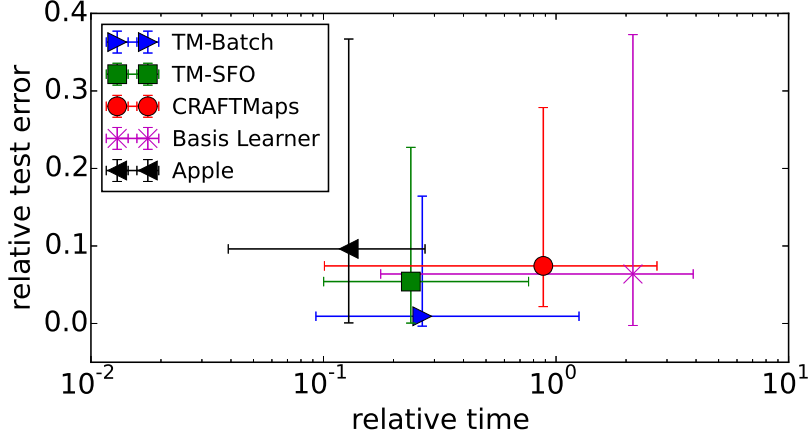


Fig. 1. The relative test error and relative running time (relative to kernel ridge regression on a subset of the training data) of several polynomial learning algorithms over 10 datasets. The median and first and third quartiles of the error and running times are shown. TM-Batch and TM-SFO are solvers for the Tensor Machine model class introduced in this work. See Section 6 for further discussion.

a regularized low-rank decomposition of a hidden tensor corresponding to the target polynomial. Given n training points in d -dimensional input space, the computational cost of fitting a degree q TM with a rank- r_{TM} approximation for each degree (using a first-order algorithm) is $\mathcal{O}(ndq^2r_{\text{TM}})$, while the cost of fitting a predictor using the Kar–Karnick random feature maps and kernel regression approach of [7,6] is $\mathcal{O}(ndqr_{\text{KK}} + nr_{\text{KK}}^2)$. In practice the r_{KK} required for accurate predictions is typically on the order of thousands; by way of comparison, we show empirically that TMs typically require less than $r_{\text{TM}} = 5$ to achieve a comparable approximation error!

We show experimentally that Tensor Machines strike a favorable balance between expressivity and parsimony, and provide an analysis of Tensor Machines that establishes favorable generalization properties. The objective function for fitting Tensor Machines is nonconvex, but we show empirically that it is sufficiently structured that the process of fitting a TM is robust, efficient, and requires very few features. We demonstrate that our algorithm exhibits a more favorable time–accuracy tradeoff when compared to the random feature map approach to polynomial regression [7,13,6], as well as to the polynomial network algorithm of [9], and the recently introduced Apple algorithm for online sparse polynomial regression [1].

2 Prior Work

Consider the estimation problem of fitting a polynomial function f to n i.i.d. training points (\mathbf{x}_i, y_i) drawn from the same unknown distribution, formulated

as

$$\hat{f} = \arg \min_{f \in \mathcal{H}_q} \frac{1}{n} \sum_{i=1}^n \ell(f(\mathbf{x}_i), y_i) + \lambda \|f\|_{\mathcal{H}_q}, \quad (1)$$

where \mathcal{H}_q is the reproducing kernel Hilbert space of all polynomials of degree at most q and ℓ is a specified loss function. An exact solution (to within a specified numerical precision) to this problem can be obtained in $\mathcal{O}(n^3)$ time using classical kernel methods.

One approach in the literature has been to couple kernel methods with various techniques for approximating the kernel matrix. The underlying assumption is that the kernel matrix is numerically low-rank, with rank $r \ll n$; typically these methods reduce the cost of kernel methods from $\mathcal{O}(n^3)$ to $\mathcal{O}(nr^2)$. Nyström approximations, sparse greedy approximations, and incomplete Cholesky factorizations fall into this class [21,19,5]. In [3], Bach and Jordan observed that prior algorithms in this class did not exploit *all* the knowledge inherent in supervised learning tasks: namely, they did not exploit knowledge of the targets (classification labels or regression values). They showed that by exploiting knowledge of the target, one can construct low-rank approximations to the kernel matrix that have significantly smaller rank, with a computational cost that remains $\mathcal{O}(nr^2)$.

Another approach to polynomial-based supervised learning relies upon the modeling assumption that the desired target function can be approximated well in the subspace of \mathcal{H}_q consisting of *sparse* polynomials. Recall that a sparse degree- q polynomial in d variables is one in which only a few of the possible monomials have nonzero coefficients (there are exponentially in d many such monomials). The early work of Sanger et al. attempts to learn the monomials relevant to the target in an online manner by augmenting the current polynomial with interaction features [18]. The recent Apple algorithm of Agarwal et al. attempts to learn a sparse polynomial, also in an online manner, using a different heuristic that selects monomials from the current polynomial to be used in forming the next term of the monomial [1]. The algorithms presented in [2] and [8] provide theoretical guarantees for fitting sparse polynomials, but their computational costs scale undesirably for large-scale learning.

Polynomial fitting has also been tackled using the neural network paradigm. In [9], Livni et al. provide an algorithm for learning polynomial functions as deep neural networks which has the property that the training error is guaranteed to decrease at each iteration. This algorithm has the desirable properties that the network learns a linear combination of a set of polynomials that are constructed in a target-dependent way, and that the degree of the polynomial does not have to be specified in advance: instead, additional layers can be added to the network until the desired error threshold has been reached, with each layer increasing the degree of the predictor by one. Unfortunately, this algorithm requires careful tuning of the hyperparameters (number of layers, and the width of each layer). In the subsequent work [10], Livni et al. provide an algorithm for fitting cubic polynomials in an iterative manner using rank-one tensor approximations. It can be shown that, in fact, this algorithm greedily fits cubic polynomials in the Tensor Machine class we propose in this paper.

Factorization Machines combine the expressivity of polynomial regression with the ability of factorization models to infer relationships from sparse training data [15]. Quadratic Factorization Machines, as introduced by Rendle, are models of the form

$$f(\mathbf{x}) = \omega_0 + \langle \boldsymbol{\omega}_1, \mathbf{x} \rangle + \sum_{i=1}^d \sum_{j=i+1}^d \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j,$$

where a vector \mathbf{v}_i is learned for each coordinate of the input \mathbf{x} . These models have been applied with great success in recommendation systems and related applications with sparse input \mathbf{x} . Quadratic FMs can be fit in time linear in the size of the data, the degree of the polynomial being fit, and the length of the vectors \mathbf{v}_i , but can only represent nonlinear interactions that can be written as symmetric homogeneous polynomials (plus a constant). In particular, FMs cannot represent polynomials that involve monomials containing variables raised to powers higher than one. For instance, x_2^2 cannot be represented as a Factorization Machine. Another drawback to Factorization Machines is that explicitly evaluating the sums involved for a degree- q FM requires $\mathcal{O}(d^q)$ operations. A computational manipulation allows quadratic FMs to be fit in linear time, and it has been claimed that similar manipulations for higher-order FMs ([15]), but no such generalizations have been documented. Perhaps for this reason, only second-order FMs have been used in the literature.

The random feature maps approach to polynomial-based learning [14] exploits the concentration of measure phenomenon to directly construct a random low-dimensional feature map that approximately decomposes \mathbf{K} . Kar and Karnick provided the first random feature map approach to polynomial regression in [7]; their key observation is that the degree- q homogeneous polynomial kernel $k(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^q$ can be approximated with random features of the form

$$\phi_i(\mathbf{x}) = \prod_{j=1}^q \langle \boldsymbol{\omega}_j^i, \mathbf{x} \rangle,$$

where the vectors $\boldsymbol{\omega}_j^i$ are vectors of random signs; that is, random feature maps of the form

$$\boldsymbol{\phi} = \frac{1}{\sqrt{r}}(\phi_1, \dots, \phi_r)$$

satisfy the condition $k(\mathbf{x}, \mathbf{y}) = \mathbb{E} [\boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{y})]$ necessary for the random feature map approach. Pagh and Pham provided a qualitatively different random feature map for polynomial kernel machines, based on the tensorization of fast hashing transforms [13]. These TensorSketch feature maps often outperform Kar–Karnick feature maps, but both methods require a large r . The CRAFTMaps approach of [6] combines either TensorSketch or Kar–Karnick random feature maps with fast Johnson–Lindenstrauss transforms to significantly reduce the number of features needed for good performance.

Of these methods, the Tensor Machine approach introduced in this paper is most similar to the Factorization Machine and Kar–Karnick random feature map approaches to polynomial learning.

3 Tensors and polynomials

To motivate Tensor Machines, which are introduced in the next section, we first review the connection between polynomials and tensors. For simplicity we consider only *homogeneous* degree- q polynomials: the monomials of such a polynomial all have degree q . We show that Kar–Karnick predictors and Factorization Machines correspond to specific tensor decompositions.

Recall that a tensor \mathbf{T} is a multidimensional array,

$$\mathbf{T} = (T_{i_1, i_2, \dots, i_q})_{i_1, \dots, i_q}.$$

The number of indices into the array, q , is also called the degree of the tensor. The inner product of two conformal tensors \mathbf{T} and \mathbf{S} is obtained by treating them as two vectors:

$$\langle \mathbf{T}, \mathbf{S} \rangle = \sum_{i_1} \sum_{i_2} \cdots \sum_{i_q} T_{i_1, i_2, \dots, i_q} S_{i_1, i_2, \dots, i_q}.$$

The Segre outer product of vectors $\boldsymbol{\omega}_1 \in \mathbb{R}^{d_1}, \dots, \boldsymbol{\omega}_q \in \mathbb{R}^{d_q}$ is the $d_1 \times d_2 \times \cdots \times d_q$ tensor that satisfies

$$(\boldsymbol{\omega}_1 \bullet \cdots \bullet \boldsymbol{\omega}_q)_{i_1, i_2, \dots, i_q} = (\boldsymbol{\omega}_1)_{i_1} (\boldsymbol{\omega}_2)_{i_2} \cdots (\boldsymbol{\omega}_q)_{i_q}.$$

The degree- q self-outer product of a vector $\boldsymbol{\omega}$ is denoted by $\boldsymbol{\omega}^{(q)}$.

Given a decomposition for a tensor \mathbf{T} of the form

$$\mathbf{T} = \sum_{i=1}^r \boldsymbol{\omega}_1^i \bullet \cdots \bullet \boldsymbol{\omega}_q^i$$

where r is minimal, r is called the rank of the tensor. A tensor \mathbf{T} is *supersymmetric* if there exists a decomposition of the form

$$\mathbf{T} = \sum_{i=1}^r \boldsymbol{\omega}_i^{(q)}.$$

The tensor $\mathbf{x}^{(q)}$ comprises all the degree- q monomials in the variable \mathbf{x} , so any degree- q homogeneous polynomial f satisfies $f(\mathbf{x}) = \langle \mathbf{T}, \mathbf{x}^{(q)} \rangle$ for some degree- q tensor \mathbf{T} ; likewise, any degree- q tensor \mathbf{T} determines a homogeneous polynomial f of degree q . This equivalence between homogeneous polynomials and tensors allows us to attack the problem of polynomial learning as one of learning a tensor. The Kar–Karnick random feature maps and Factorization Machine approaches can both be viewed through this lens.

A single Kar–Karnick random polynomial feature corresponds to a rank-one tensor:

$$\begin{aligned}\phi(\mathbf{x}) &= \prod_{j=1}^q \langle \boldsymbol{\omega}_j, \mathbf{x} \rangle = \prod_{j=1}^q \left(\sum_{i=1}^d (\boldsymbol{\omega}_j)_i x_i \right) \\ &= \sum_{i_1=1}^d \cdots \sum_{i_q=1}^d (\boldsymbol{\omega}_1)_{i_1} \cdots (\boldsymbol{\omega}_q)_{i_q} x_{i_1} \cdots x_{i_q} \\ &= \langle \boldsymbol{\omega}_1 \bullet \cdots \bullet \boldsymbol{\omega}_q, \mathbf{x}^{(q)} \rangle.\end{aligned}$$

Accordingly, predictors generated by the random feature maps approach with the Kar–Karnick feature map ϕ satisfy

$$f(\mathbf{x}) = \sum_{i=1}^r \alpha_i \phi_i(\mathbf{x}) = \left\langle \sum_{i=1}^r \alpha_i \boldsymbol{\omega}_1^i \bullet \boldsymbol{\omega}_2^i \bullet \cdots \bullet \boldsymbol{\omega}_q^i, \mathbf{x}^{(q)} \right\rangle.$$

for some coefficient vector $\boldsymbol{\alpha}$. That is, Kar–Karnick predictors correspond to tensors in the span of r *randomly sampled* degree- q rank-one tensors.

Factorization Machines fit predictors of the form

$$f(\mathbf{x}) = \sum_{i_q > i_{q-1} > \cdots > i_1} \langle \mathbf{v}_{i_1}, \dots, \mathbf{v}_{i_q} \rangle x_{i_1} \cdots x_{i_q},$$

where the factors \mathbf{v}_i are vectors in \mathbb{R}^m and

$$\langle \mathbf{v}_{i_1}, \dots, \mathbf{v}_{i_q} \rangle = \sum_{j=1}^m (\mathbf{v}_{i_1})_j \cdots (\mathbf{v}_{i_q})_j$$

is the generalization of the inner product of two vectors. Let \mathbf{V} be the $d \times m$ matrix with rows comprising the vectors \mathbf{v}_i , and define the d -dimensional vector $\boldsymbol{\omega}_j$ to be the j th column of \mathbf{V} . It follows that FMs can be expressed in the form

$$\begin{aligned}f(\mathbf{x}) &= \sum_{i_1=1}^{d-q+1} \cdots \sum_{i_q=i_{q-1}+1}^d \left(\sum_{j=1}^m (\mathbf{v}_{i_1})_j \cdots (\mathbf{v}_{i_q})_j \right) x_{i_1} \cdots x_{i_q} \\ &= \sum_{i_1=1}^{d-q+1} \cdots \sum_{i_q=i_{q-1}+1}^d \left(\sum_{j=1}^m (\boldsymbol{\omega}_j)_{i_1} \cdots (\boldsymbol{\omega}_j)_{i_q} \right) x_{i_1} \cdots x_{i_q} \\ &= \sum_{i_1=1}^{d-q+1} \cdots \sum_{i_q=i_{q-1}+1}^d \left(\sum_{j=1}^m \boldsymbol{\omega}_j^{(q)} \right)_{i_1, \dots, i_q} \left(\mathbf{x}^{(q)} \right)_{i_1, \dots, i_q} \\ &= \left\langle \sum_{j=1}^m \boldsymbol{\omega}_j^{(q)} - \mathbf{R}, \mathbf{x}^{(q)} \right\rangle,\end{aligned}$$

where the tensor \mathbf{R} is defined so that its nonzero entries cancel the entries of $\sum_{j=1}^m \omega_j^{(q)}$ that correspond to coefficients of monomials like $x_1 x_2^2$ that contain a variable raised to a power larger than one. The vectors ω_j are learned during training (the tensor \mathbf{R} is implicitly defined by these vectors).

4 Tensor Machines

Thus, the Kar–Karnick random feature map approach searches for a low-rank tensor corresponding to the target polynomial and the Factorization Machine approach attempts to provide a supersymmetric low-rank tensor plus sparse tensor decomposition of the target polynomial. The Kar–Karnick approach has the drawback that it tries to find a good approximation in the span of a set of random rank-one tensors, so many such basis elements must be chosen to ensure that an unknown target can be represented well. Factorization Machines circumvent this issue by directly learning the tensor decomposition, but impose strict constraints on the form of the fitted polynomial that are not appropriate for general learning applications.

As an alternative to Kar–Karnick random feature maps and factorization machines, we propose to instead fit Tensor Machines by *directly optimizing* over rank-one tensors to directly form a low-rank approximation to the target tensor (i.e., the target polynomial). Since the targets are not, in general, homogenous polynomials, we learn a different set of rank-one factors for each degree up to q .

More precisely, Tensor Machines are functions of the form

$$f(\mathbf{x}) = \omega^0 + \langle \omega^1, \mathbf{x} \rangle + \sum_{p=2}^q \left\langle \sum_{i=1}^r \omega_1^{p,i} \bullet \dots \bullet \omega_p^{p,i}, \mathbf{x}^{(p)} \right\rangle,$$

obtained as minimizers to the following proxy to the kernel machine objective (1):

$$\frac{1}{n} \sum_{i=1}^n \ell(f(\mathbf{x}_i), y_i) + \lambda \|\omega^1\|_2^2 + \lambda \sum_{p=2}^q \sum_{i=1}^r \sum_{j=1}^p \|\omega_j^{p,i}\|_2^2. \quad (2)$$

By construction, Tensor Machines couple the expressiveness of the Kar–Karnick random feature maps model with the parsimony of Factorization Machines.

5 Generalization Error

In this section we argue that fitting Tensor Machines using empirical risk minimization makes efficient use of the training data. We show that the observed risk of Tensor Machines converges to the expected risk at the optimal rate, thus indicating that empirical risk minimization is an efficient method for finding locally optimal Tensor Machines (assuming the optimization method avoids saddle points). For convenience, we consider a variant of Tensor Machines where the

norms of the vectors $\omega_j^{p,\ell}$ are constrained:

$$\begin{aligned} & \text{minimize} \quad \frac{1}{n} \sum_{i=1}^n \ell \left(\omega^0 + \langle \omega^1, \mathbf{x}_i \rangle + \sum_{p=2}^q \langle \mathbf{T}_p, \mathbf{x}_i^{(p)} \rangle, y_i \right) \\ & \text{subject to} \quad \mathbf{T}_p = \sum_{j=1}^r \omega_1^{p,j} \bullet \cdots \bullet \omega_p^{p,j} \text{ for } p \geq 2, \\ & \quad \|\omega^1\|_2 \leq B, \\ & \quad \|\omega_i^{p,j}\|_2 \leq B \text{ for all } p, j, i. \end{aligned}$$

This formulation is not equivalent to the formulation given in (2), but by taking B to be the norm of the largest constituent vector in the fitted Tensor Machine, any bound on the generalization error of this formulation applies to the generalization error of Tensor Machines obtained by minimizing (2).

Recall that the Rademacher $\mathcal{R}_n(\mathcal{F})$ complexity of a function class \mathcal{F} measures how well random noise can be approximated using a function from \mathcal{F} [4].

Definition 1 (Rademacher complexity). *Given a function class \mathcal{F} and i.i.d. random variables \mathbf{z}_i , the Rademacher complexity of \mathcal{F} , $\mathcal{R}_n(\mathcal{F})$, is defined as*

$$\mathcal{R}_n(\mathcal{F}) = \frac{1}{n} \mathbb{E}_{\{\mathbf{z}_i\}, \sigma} \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^n \sigma_i f(\mathbf{z}_i) \right],$$

where the σ_i are independent Rademacher random variables (random variables taking values ± 1 with equal probability).

Well-known results (e.g. [4][Theorems 7 and 8]) state that, with high probability over the training data, the observed classification and regression risks are within $\mathcal{O}(\mathcal{R}_n(\mathcal{F}) + 1/\sqrt{n})$ of the true classification and regression risks. In fact, the optimal rate of convergence for the observed risk to the expected risk is $\mathcal{O}(1/\sqrt{n})$, which can only be achieved when the Rademacher complexity of the hypothesis class is $\mathcal{O}(1/\sqrt{n})$ [11].

Our main observation is that the Rademacher complexity of a Tensor Machine grows at the rate $\mathcal{O}(1/\sqrt{n})$, so the empirically observed estimate of the Tensor Machine risk converges to the expected risk at the optimal rate.

Theorem 1. *Let $\mathcal{F}_{d,q,r,B}$ denote the class of degree- q , rank- r Tensor Machines on \mathbb{R}^d with constituent vectors constrained to lie in the ball of radius B :*

$$\begin{aligned} \mathcal{F}_{d,q,r,B} = & \left\{ f : \mathbf{x} \mapsto \omega^0 + \langle \omega^1, \mathbf{x} \rangle + \sum_{p=2}^q \langle \mathbf{T}_p, \mathbf{x}^{(p)} \rangle \mid \right. \\ & \mathbf{T}_p = \sum_{j=1}^r \omega_1^{p,j} \bullet \cdots \bullet \omega_p^{p,j} \text{ for } p \geq 2, \\ & \left. \|\omega^1\|_2 \leq B, \text{ and } \|\omega_i^{p,j}\|_2 \leq B \text{ for all } p, j, i \right\}. \end{aligned}$$

Let (\mathbf{x}, y) be distributed according to \mathbb{P} , and assume $\|\mathbf{x}\|_2 \leq B_{\mathbf{x}}$ almost surely. The Rademacher complexity, with respect to (\mathbf{x}, y) , of this hypothesis class satisfies

$$\mathcal{R}_n(\mathcal{F}_{d,q,r,B}) \leq \frac{cr(1 + 8BB_{\mathbf{x}})^q q^2 (\sqrt{qd \ln d} + \sqrt{d})}{\sqrt{n}}$$

where c is a constant.

This result follows from reformulating the Rademacher complexity as the spectral norm of a Rademacher sum of data-dependent tensors and then applying a recent bound from [12] on the spectral norm of random tensors. A full proof is provided in the appendix.

6 Empirical Evaluations

In this section, we evaluate the performance of Tensor Machines on several real-world regression and classification datasets and demonstrate their attractiveness in learning polynomial features relative to other state-of-the-art algorithms.

6.1 Experimental setup

The nonconvex optimization problem (2) is central to fitting Tensor Machines. We consider two solvers for (2). The first uses the implementation of L-BFGS provided in Mark Schmidt’s `minFunc`³ MATLAB package for optimization [16]. Since L-BFGS is a batch optimization algorithm, we also investigate the use of SFO, a stochastic quasi-Newton solver designed to work with minibatches [20]. We use the reference implementation of SFO provided by Sohl-Dickstein⁴. We refer to the two algorithms used to fit Tensor Machines as TM-Batch and TM-SFO, respectively.

The choice of initialization strongly influences the performance of both TM-Batch and TM-SFO. Accordingly, we used initial points sampled from a $\mathcal{N}(0, \alpha^2)$ distribution. The variance α^2 as well as the regularization parameter λ in (2) are set by tuning.

We compared TM-Batch and TM-SFO against several recent algorithms for learning polynomials: CRAFTMaps [6], Basis Learner [9] and Apple [1]. To provide a baseline, we also used Kernel Ridge Regression (KRR) to learn polynomials; in cases where the training set contained more than 40,000 points, we randomly chose a subset of size 40,000 to perform KRR. For CRAFTMaps, the up-projection dimensionality is set to be 4 times the down-projection dimensionality. We did not obtain reasonable predictions using the original implementation of Apple in `vowpal wabbit`, so we implemented the model in MATLAB and used this implementation to train and test the model, but reported the running time of the VW implementation (with the same choice of parameters). The remaining

³ The 2012 release, retrieved from <http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html>

⁴ <https://github.com/Sohl-Dickstein/Sum-of-Functions-Optimizer/blob/master/README.md>

Name	Train/Test split	d	Type	q
Indoor	19937/11114	520	regression	3
Year	463715/51630	90	regression	2
Census	18186/2273	119	regression	2
Slice	42291/10626	384	regression	5
Buzz	466600/116650	77	regression	3
Gisette	6000/1000	5000	binary	3
Adult	32561/16281	122	binary	3
Forest	522910/58102	54	binary	4
eBay search	500000/100000	478	binary	3
Cor-rna	59535/157413	8	binary	4

Table 1. Description of datasets. The target degree q was selected by minimizing the KRR test error on each dataset (or a subset).

methods are implemented in pure MATLAB. The experiments were conducted on a machine with four 6-core Intel Xeon 2 GHz processors and 128 GB RAM.

Because the performance of nonlinear learning algorithms can be very data-dependent, we tested the methods on a collection of 10 publicly available datasets to provide a broad characterization of the behavior of these algorithms. A summary of the basic properties of these datasets can be found in Table 1. For each dataset, the target degree r was chosen to be the value that minimized the KRR test error. We preprocessed the data by first normalizing the input features to have similar magnitudes — viz., so that each column of the training matrix has unit Euclidean norm — then scaling each datapoint to have unit Euclidean norm.

For regression tasks, we used the squared loss, and for binary classification tasks we used the logistic loss. For regression tasks, the test error is reported as $\|\hat{y} - y^*\|_2 / \|y^*\|_2$ where \hat{y} and y^* are the prediction and ground truth, respectively; inaccuracy is reported for classification tasks.

Each algorithm is governed by several interacting parameters, but for each algorithm we identify one major parameter: the number of iterations for TM-Batch, the number of epochs for TM-SFO, the number of features for CRAFTMaps, the layer width for Basis Learner, and the expansion parameter for Apple. We choose optimal values for the non-major parameters through a 10-fold cross-validation on the training set. We varied the major parameter over a wide range and recorded the test error and running times for the different values until the performance of the algorithms saturated. By saturation, we mean that the improvement in the test error is less than $0.02 \times \text{err}(\mathbf{krr})$ where $\text{err}(\mathbf{krr})$ denotes the test error of KRR. For each combination of parameters, the average test errors and running times over 3 independent trials are reported.

6.2 Overall performance

To compare their performance, we applied TM-Batch, TM-SFO, CRAFTMaps, Basis Learner, and Apple to the 10 datasets listed in Table 1. We did not evaluate Factorization Machines because efficient algorithms for fitting FM models are only available in the literature for the case $q = 2$. It is unclear whether FMs with higher values of q can be fit efficiently.

We present the computation/prediction accuracy tradeoffs of the considered algorithms in Figure 1. The data plotted are the test errors and running times of the algorithms relative to those of KRR:

$$\text{relerr} = (\text{err}(\text{alg}) - \text{err}(\text{krr}))/\text{err}(\text{krr})$$

and

$$\text{reltime} = \text{time}(\text{alg})/\text{time}(\text{krr}).$$

The median values of relerr and reltime as well as the corresponding first and third quartiles are shown.

The detailed results can be found in Table 2. The performance of TM-Batch is consistent across the datasets in the sense that it provides reliable predictions in a fairly short amount of time. TM-SFO converges to lower quality solutions than TM-Batch, but has lower median and variance in runtime. CRAFTMaps and Basis Learner take significantly more time to yield solutions almost as accurate as the two Tensor Machine algorithms. As one might expect, due to its greedy nature, Apple is the fastest of the algorithms (the time of the `vowpal wabbit` implementation is reported), but of all the algorithms, TM-Batch and TM-SFO deliver the lowest worst-case relative errors.

Name	KRR	TM-Batch	TM-SFO	CRAFTMaps	Basis Learner	Apple
Indoor	0.00961/50	0.00802/3.2(4)	0.0182/3.9(4)	0.00559/2.7(300)	0.00241 /97	0.00712/6.5
Year	0.00485/310	0.00565/85(5)	0.00484 /74(5)	0.00494/39(200)	0.00496/67	0.00489/60
Census	0.0667/46	0.0774/12(5)	0.0802/45(5)	0.0767 /7.3(1000)	0.0788/8.1	0.0950/7.3
Slice	0.0118/441	0.0229/80(3)	0.0213 /73(3)	0.0465/707(7000)	0.0501/1034	0.0788/46
Buzz	0.407/624	0.407/568(2)	0.409/440(2)	0.408/2105(1200)	0.373 /2472	0.496/325
Gisette	0.027/6.6	0.0266/16(4)	0.0254/29(4)	0.0300/63(11500)	0.0270/62	0.0240 /32
Adult	0.150/182	0.149 /0.9(4)	0.151/2.0(4)	0.154/17(700)	0.149/30	0.150/0.8
Forest	0.148/361	0.184/657(5)	0.178 /569(5)	0.196/1023(750)	0.180/3494	0.219/14
eBay search	0.197/446	0.197/612(5)	0.192 /349(5)	0.281/1054(800)	0.281/1642	0.269/122
Cor-rna	0.0446/423	0.0453 /8.4(3)	0.0489/7.2(3)	0.0462/19(200)	0.0493/1.7	0.0489/4.2

Table 2. Test error/running time of each methods on 10 datasets. The rank parameter r used in TM-Batch and TM-SFO and number of features used in CRAFTMaps is listed in the parenthesis. For Apple, the test error is computed based on a MATLAB implementation of Apple but the running time is recorded by using the `vowpal wabbit` framework with the same choice of parameters.

Both TM solvers required less than $r = 5$ rank-one TM features for each individual degree fit for *all* datasets. Since only one dataset benefited from a quintic fit, the TM models required at most $1 + d + \sum_{p=2}^q prd \leq 72d$ parameters

to fit each dataset; this should be compared to the CRAFTMaps models, which required at least $400d \leq qrd$ parameters to fit each dataset, generally produced fits with higher error than the TM solvers, and required longer solution times.

6.3 Effect of rank parameter r

To investigate the effect of the rank parameter r on the test error of Tensor Machines, we evaluated both TM-Batch and TM-SFO on the Census and Slice datasets. On Census, since the target degree is 2, we evaluate Factorization Machines (FM) as well. The results are shown in Figure 2.

As expected, increasing r leads to smaller test errors. On Slice, where the target degree $q = 5$, the gap between the performance of TMs and KRR is relatively large (almost a factor of 2), while on Census, where $q = 2$, the gap is slighter. Interestingly, on Census, increasing rank does not lead to a higher accuracy after $r \geq 3$. We also observe on Census that the error of TMs is lower than that of FMs; this behaviour was also observed on the other datasets when $q = 2$.

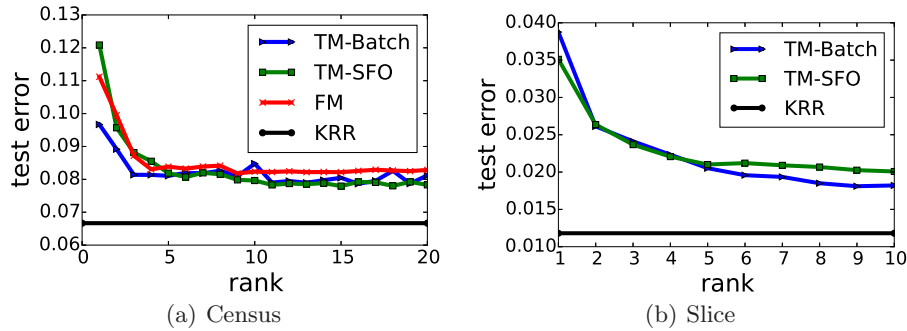


Fig. 2. Test error of TM-Batch and TM-SFO with different rank parameters r on the Census ($q = 2$) and Slice ($q = 5$) datasets. The test error of using kernel ridge regression (KRR) is also plotted. On Census, we also evaluate Factorization Machines (FM).

6.4 Time-accuracy tradeoffs

To assess the time-accuracy tradeoffs of the various algorithms, in Figure 3 we plot the test error vs the training time for the Slice and Forest datasets. The training time is determined by varying the settings of each algorithm’s major parameter.

TM-Batch and TM-SFO compare favorably against the other methods: they either produce a much lower error than the other methods (on Slice) or reach a considerably low error much faster (on Forest). Similar patterns were observed on most of the datasets we considered.

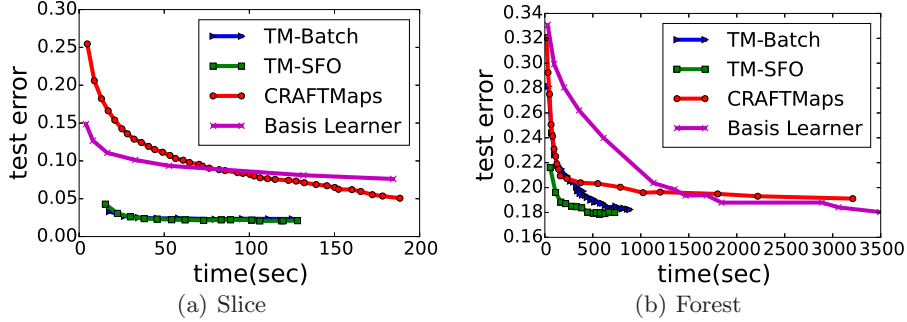


Fig. 3. Evaluation of time-accuracy tradeoffs of the algorithms on the Slice ($q = 5$) and Forest ($q = 4$) datasets. The rank parameters r for TM-Batch and TM-SFO on Slice and Forest are 3 and 5, respectively.

6.5 Scalability of TM-SFO on eBay search dataset

Since SFO needs to access only a small mini-batch of data points per update, it is suitable for fitting TMs on datasets which cannot fit in memory all at once. Here we explore the scalability of TM-SFO on a private eBay search dataset of 1.2 million data points with dimension 478. Each data point (u_a, u_b) comprises the feature vectors associated with a pair of items that were returned as the results of a search on the eBay website and were subsequently visited. The goal is to fit a polynomial for the task of classifying which item was clicked on first: a or b . In our experiments, we randomly selected 100,000 data points to be the test set; training sets of variable size were selected from the remainder.

For comparison, we also evaluate CRAFTMaps on the same task. In TM-SFO, we fix the rank parameter r to be 5 and number of epochs to be 50. In CRAFTMaps, we fix the number of random features to be 800. These parameters are the optimal settings from the experiments used to generate Table 2. We report both the classification error and training time as the size of training set grows.

The results are shown in Figure 4. We observe that the running time of TM-SFO grows almost linearly with the size of the training set, while that of CRAFTMaps grows superlinearly. Also, as expected, because CRAFTMaps choose the hypothesis space independently of the target, increasing the size of the training set without also increasing the size of the model does not give much gain in performance past 100,000 training points. We see that the target-adaptive nature of TMs endows TM-SFO with two advantages. First, for a fixed amount of training data the errors of TMs are significantly lower than those of CRAFTMaps. Second, because the hypothesis space evolves as a function of the target and training data, the training error of TM-SFO exhibits noticeable decay up to a training set size of about 500,000 points, long past the point where the CRAFTMaps test error has saturated.

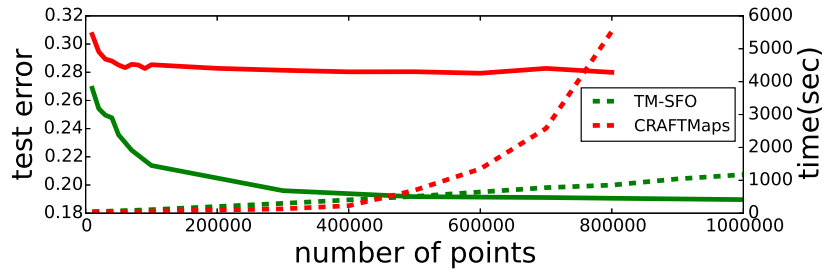


Fig. 4. Scalability of TM-SFO on eBay search dataset ($q = 3$, $r = 5$). The number of epochs is set to be 50. The plot shows the test error (black solid line) and training time (red dash line) as a function of the number of training point used.

References

1. A. Agarwal, A. Beygelzimer, D. Hsu, J. Langford, and M. Telgarsky. Scalable Nonlinear Learning with Adaptive Polynomial Expansions. In *Advances in Neural Information Processing Systems 27*, 2014.
2. A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang. Learning sparse polynomial functions. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2014.
3. F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.
4. P. L. Bartlett and S. Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research*, 2002.
5. S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2002.
6. R. Hamid, Y. Xiao, A. Gittens, and D. DeCoste. Compact Random Feature Maps. In *Proceedings of The 31st International Conference on Machine Learning*, 2013.
7. P. Kar and H. Karnick. Random Feature Maps for Dot Product Kernels. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, 2012.
8. M. Kocaoglu, K. Shanmugam, A. G. Dimakis, and A. Klivans. Sparse Polynomial Learning and Graph Sketching. In *Advances in Neural Information Processing Systems 27*, 2014.
9. R. Livni, S. Shalev-Shwartz, and O. Shamir. An algorithm for learning polynomials. Preprint arXiv:1304.7045, 2014.
10. R. Livni, S. Shalev-Shwartz, and O. Shamir. On the Computational Efficiency of Training Neural Networks. In *Advances in Neural Information Processing Systems 27*, 2014.
11. S. Mendelson. Lower bounds for the empirical minimization algorithm. *IEEE Transactions on Information Theory*, 2008.
12. N. H. Nguyen, P. Drineas, and T. D. Tran. Tensor sparsification via a bound on the spectral norm of random tensors. *Information and Inference*, abs/1005.4732, 2013.

13. N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013.
14. A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems 20*, 2007.
15. S. Rendle. Factorization machines. In *IEEE 10th International Conference on Data Mining*, 2010.
16. B. H. Richard, N. Jorge, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63, 1994.
17. W. Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 3 edition, 1976.
18. T. D. Sanger, R. S. Sutton, and C. J. Matteus. Iterative Construction of Sparse Polynomial Approximation. In *Advances in Neural Information Processing Systems*, 1992.
19. A. J. Smola and B. Schölkopf. Sparse Greedy Matrix Approximation for Machine Learning. In *Proceedings of the 17th International Conference on Machine Learning*, 2000.
20. J. Sohl-Dickstein, B. Poole, and S. Ganguli. Fast large-scale optimization by unifying stochastic gradient and quasi-Newton methods. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.
21. C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, 2001.
22. J. Yang, V. Sindhwani, H. Avron, and M. Mahoney. Quasi-Monte Carlo Feature Maps for Shift-Invariant Kernels. In *Proceedings of The 31st International Conference on Machine Learning*, 2014.
23. J. Yang, V. Sindhwani, Q. Fan, H. Avron, and M. Mahoney. Random laplace feature maps for semigroup kernels on histograms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
24. Z. Yang, A. J. Smola, L. Song, and A. G. Wilson. À la carte — learning fast kernels. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015.
25. F. X. Yu, S. Kumar, H. Rowley, and S.-F. Chang. Compact Nonlinear Maps and Circulant Extensions. Preprint arXiv:1503.03893, 2015.

A Proof of Theorem 1

Theorem 1 is a corollary of the following bound on the Rademacher complexity of a rank-one Tensor Machine.

Theorem 2. *Let $\mathcal{F}_{d,q,B}$ denote the class of degree- q rank-one Tensor Machines on \mathbb{R}^d with constituent vectors constrained to lie in the ball of radius B :*

$$\mathcal{F}_{d,q,B} = \left\{ f : \mathbf{x} \mapsto \left\langle \boldsymbol{\omega}_1 \bullet \cdots \bullet \boldsymbol{\omega}_q, \mathbf{x}^{(q)} \right\rangle \mid \|\boldsymbol{\omega}_j\|_2 \leq B \text{ for } j = 1, \dots, q \right\}$$

Let (\mathbf{x}, y) be distributed according to \mathbb{P} , and assume $\|\mathbf{x}\|_2 \leq B_{\mathbf{x}}$ almost surely. The Rademacher complexity with respect to (\mathbf{x}, y) of this hypothesis class satisfies

$$\mathcal{R}_n(\mathcal{F}_{d,q,B}) \leq \frac{c(8BB_{\mathbf{x}})^q q(\sqrt{qd \ln d} + \sqrt{d})}{\sqrt{n}},$$

where c is a constant.

Proof (Proof of Theorem 1). Every function in $\mathcal{F}_{d,q,r,B}$ can be written as the sum of a constant, a linear function of the form $\mathbf{x} \mapsto \langle \boldsymbol{\omega}, \mathbf{x} \rangle$ with $\|\boldsymbol{\omega}\| < B$, and r functions from each of $\mathcal{F}_{d,2,B}, \dots, \mathcal{F}_{d,q,B}$. The Rademacher complexity of \mathbb{R} is zero, and a straightforward calculation establishes that the Rademacher complexity of linear functions of the specified form is no larger than $BB_{\mathbf{x}}/\sqrt{n}$. It follows from a simple structural result on Rademacher complexities [4][Theorem 12] that

$$\mathcal{R}_n(\mathcal{F}) \leq \frac{BB_{\mathbf{x}}}{\sqrt{n}} + s \sum_{p=2}^q \mathcal{R}_n(\mathcal{F}_{d,p,B}).$$

Applying Theorem 2, we have that

$$\begin{aligned} \mathcal{R}_n(\mathcal{F}) &\leq \frac{BB_{\mathbf{x}}}{\sqrt{n}} + cr \sqrt{\frac{d \ln d}{n}} \sum_{p=2}^q (8BB_{\mathbf{x}})^p p^{3/2} + cr \sqrt{\frac{d}{n}} \sum_{p=2}^q (8BB_{\mathbf{x}})^p p \\ &\leq cr \sqrt{\frac{d \ln d}{n}} (1 + 8BB_{\mathbf{x}})^q \sum_{p=1}^q p^{3/2} + cr \sqrt{\frac{d}{n}} (1 + 8BB_{\mathbf{x}})^q \sum_{p=1}^q p \\ &\leq \frac{cr(1 + 8BB_{\mathbf{x}})^q q^2 (\sqrt{qd \ln d} + \sqrt{d})}{\sqrt{n}}. \end{aligned}$$

Proof (Proof of Theorem 2). Given the data points $\mathbf{x}_1, \dots, \mathbf{x}_n$, let

$$\hat{\mathcal{R}}_n(\mathcal{F}) = \frac{1}{n} \mathbb{E}_{\sigma} \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^n \sigma_i f(\mathbf{x}_i) \right]$$

so that $\mathcal{R}_n(\mathcal{F}) = \mathbb{E}_{\{\mathbf{x}_1, \dots, \mathbf{x}_n\}} \hat{\mathcal{R}}_n(\mathcal{F})$. From the definition of $\mathcal{F}_{d,q,B}$, we have that

$$\begin{aligned} \hat{\mathcal{R}}_n(\mathcal{F}) &= \frac{1}{n} \mathbb{E}_{\sigma} \left[\sup_{\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_q \in \Omega_B} \sum_{i=1}^n \sigma_i \langle \boldsymbol{\omega}_1 \bullet \dots \bullet \boldsymbol{\omega}_q, \mathbf{x}_i^{(q)} \rangle \right] \\ &= \frac{1}{n} \mathbb{E}_{\sigma} \left[\sup_{\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_q \in \Omega_B} \left\langle \boldsymbol{\omega}_1 \bullet \dots \bullet \boldsymbol{\omega}_q, \sum_{i=1}^n \sigma_i \mathbf{x}_i^{(q)} \right\rangle \right] \end{aligned}$$

where $\Omega_B = \{\boldsymbol{\omega} \in \mathbb{R}^d \mid \|\boldsymbol{\omega}\|_2 \leq B\}$.

Define $\mathbf{T}_{\sigma} = \sum_{i=1}^n \sigma_i \mathbf{x}_i^{(q)}$. Recall the definition of the spectral norm of an order- q tensor \mathbf{T} :

$$\|\mathbf{T}\| = \sup_{\mathbf{v}_1, \dots, \mathbf{v}_q \in \Omega_1} |\langle \mathbf{T}, \mathbf{v}_1 \bullet \dots \bullet \mathbf{v}_q \rangle|.$$

It follows from this definition that

$$\begin{aligned} \hat{\mathcal{R}}_n(\mathcal{F}) &\leq \frac{1}{n} \mathbb{E}_{\sigma} \left[B^q \cdot \sup_{\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_q \in \Omega_B} \left\langle \mathbf{T}_{\sigma}, \frac{\boldsymbol{\omega}_1}{\|\boldsymbol{\omega}_1\|_2} \bullet \dots \bullet \frac{\boldsymbol{\omega}_q}{\|\boldsymbol{\omega}_q\|_2} \right\rangle \right] \\ &= \frac{1}{n} B^q \mathbb{E}_{\sigma} [\|\mathbf{T}_{\sigma}\|]. \end{aligned} \tag{3}$$

For simplicity, we drop the subscript σ from \mathbf{T}_σ in the following. Note that $\mathbb{E}[\mathbf{T}] = 0$, since Rademacher variables are mean-zero. We now apply Theorem 2 of [12], which bounds the expected spectral norm of the difference between a tensor Rademacher sum and its expectation with the sum of the expected maximum Euclidean lengths of one-dimensional slices through the tensor:

$$\begin{aligned}\mathbb{E}_\sigma [\|\mathbf{T}\|] &= \mathbb{E}_\sigma [\|\mathbf{T} - \mathbb{E}\mathbf{T}\|] \\ &\leq c8^q(\sqrt{q \ln d} + 1) \left(\sum_{j=1}^q \mathbb{E}_\sigma \left[\max_{i_1, \dots, i_{j-1}, i_{j+1}, \dots, i_q} \left(\sum_{i_j=1}^d T_{i_1, \dots, i_q}^2 \right)^{\frac{1}{2}} \right] \right).\end{aligned}$$

We replace the innermost sum with a maximum to obtain an estimate involving the expected size of the largest entry in \mathbf{T} :

$$\begin{aligned}\mathbb{E}_\sigma [\|\mathbf{T}\|] &\leq c8^q(\sqrt{q \ln d} + 1) \left(\sum_{j=1}^q \mathbb{E}_\sigma \left[\max_{i_1, \dots, i_{j-1}, i_{j+1}, \dots, i_q} (d \max_{i_j} T_{i_1, \dots, i_q}^2)^{\frac{1}{2}} \right] \right) \\ &= c8^q q(\sqrt{qd \ln d} + \sqrt{d}) \mathbb{E}_\sigma \left[\max_{i_1, \dots, i_q} |T_{i_1, \dots, i_q}| \right].\end{aligned}\quad (4)$$

Next we bound the maximum entry of \mathbf{T} with the Frobenius norm of \mathbf{T} , and apply Jensen's inequality to obtain

$$\begin{aligned}\mathbb{E}_\sigma \left[\max_{i_1, \dots, i_q} |T_{i_1, \dots, i_q}| \right] &\leq \mathbb{E}_\sigma \left[\langle \mathbf{T}, \mathbf{T} \rangle^{\frac{1}{2}} \right] \leq (\mathbb{E}_\sigma [\langle \mathbf{T}, \mathbf{T} \rangle])^{\frac{1}{2}} \\ &= \left(\sum_{i_1, \dots, i_q=1}^d \mathbb{E}_\sigma [T_{i_1, \dots, i_q}^2] \right)^{\frac{1}{2}}.\end{aligned}$$

Recalling the definition of \mathbf{T} , we have that

$$\begin{aligned}\left(\mathbb{E}_\sigma \left[\max_{i_1, \dots, i_q} |T_{i_1, \dots, i_q}| \right] \right)^2 &\leq \sum_{i_1, \dots, i_q=1}^d \mathbb{E}_\sigma \left[\left(\sum_{i=1}^n \sigma_i \mathbf{x}_i^{(q)} \right)_{i_1, \dots, i_q}^2 \right] \\ &= \sum_{i_1, \dots, i_q=1}^d \sum_{i,j=1}^n \mathbb{E}_\sigma [\sigma_i \sigma_j] (\mathbf{x}_i^{(q)})_{i_1, \dots, i_q} (\mathbf{x}_j^{(q)})_{i_1, \dots, i_q} \\ &= \sum_{i_1, \dots, i_q=1}^d \sum_{i=1}^n (\mathbf{x}_i^{(q)})_{i_1, \dots, i_q}^2 = \sum_{i=1}^n \langle \mathbf{x}_i^{(q)}, \mathbf{x}_i^{(q)} \rangle.\end{aligned}$$

It is readily established that

$$\langle \mathbf{v}_1 \bullet \dots \bullet \mathbf{v}_q, \mathbf{v}_1 \bullet \dots \bullet \mathbf{v}_q \rangle = \|\mathbf{v}_1\|^2 \dots \|\mathbf{v}_q\|^2$$

for any rank-one tensor, so it follows that

$$\mathbb{E}_\sigma \left[\max_{i_1, \dots, i_q} |T_{i_1, \dots, i_q}| \right] \leq \sqrt{\sum_{i=1}^n \|\mathbf{x}_i\|^{2q}} \leq \sqrt{n} B_{\mathbf{x}}^q. \quad (5)$$

From equations (3), (4), and (5), we conclude that

$$\mathcal{R}_n(\mathcal{F}) = \mathbb{E}_{\mathbf{x}_1, \dots, \mathbf{x}_n} \left[\hat{\mathcal{R}}_n(\mathcal{F}) \right] \leq \frac{c(8BB_{\mathbf{x}})^q q (\sqrt{qd \ln d} + \sqrt{d})}{\sqrt{n}}.$$